

# R FOR DATA SCIENCE

Research Methods in Psychology I & II • Department of Psychology • Colorado State University

## BY THE END OF THIS INTRODUCTION YOU WILL:

1. Be familiar with the field of Data Science.
2. Have R and RStudio successfully loaded on your laptop.
3. Know a bit about the key features of R Studio.

## A FEW R RESOURCES:

### R Bloggers:

<https://www.r-bloggers.com/>

### Stack Overflow (community):

<https://stackoverflow.com/>

### R for Data Science:

<http://r4ds.had.co.nz/>

### Quick R:

<https://www.statmethods.net/>

### ATS UCLA stat examples:

<https://stats.idre.ucla.edu/other/dae/>

### R Documentation

<https://www.rdocumentation.org/>

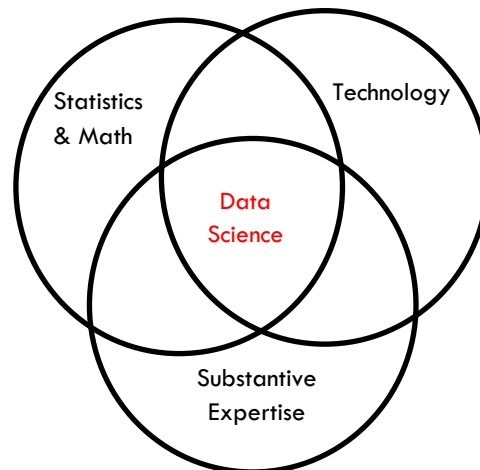
### DataCamp:

<https://www.datacamp.com/>

## What is Data Science?

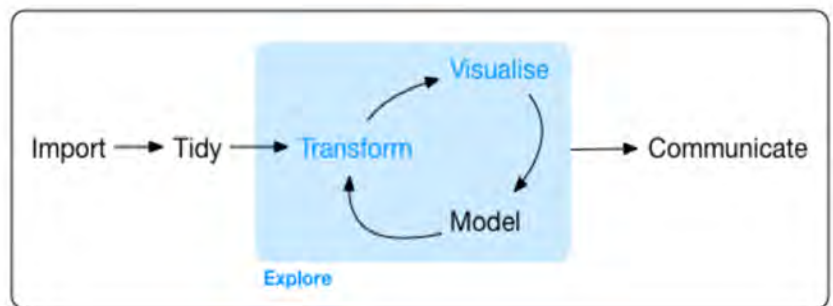
Data science is the discipline of using raw data to produce insight.

To build analytic models, we must have a strong foundation in mathematics and statistics.



To build analytic models, we must garner the power of technology.

To build analytic models, we must have a deep understanding of the phenomena at play.



Wickham & Grolemund—R for Data Science

## What is R?

R is a powerful language, computing, and graphics environment. It is made up of a base software program and thousands of packages that can be added (packages written by some of the greatest statisticians and data scientists in the world). R is available for Windows and Mac, and is both open source and free.

To install R, go to <http://www.r-project.org/> (click on the download R link under the getting started section). You will be prompted to choose a CRAN mirror — click on <https://cloud.r-project.org/>. Choose the “precompiled binary distribution” (first section) for your operating system (Windows or Mac).

If you are a Windows user, click on Download R for Windows, then click on install R for the first time, and finally click on download R for Windows. This will download a .exe file. Double click on it and follow the directions to install.

If you are a Mac user, please make sure that your operating system is up to date before you begin. Then, click on Download R for Mac OS X. Scroll down to the header Latest Release. See the note under the first paragraph that indicates you should install XQuartz. Click on this and follow the directions to download and install XQuartz. Once complete, return to the Download R for Mac OS X page. Now, click on the .pkg file listed under Latest Release. Follow the directions to download and install R.

An updated version of R comes out about once per year, with minor versions in between. It's a good idea to upgrade regularly.

## What is RStudio

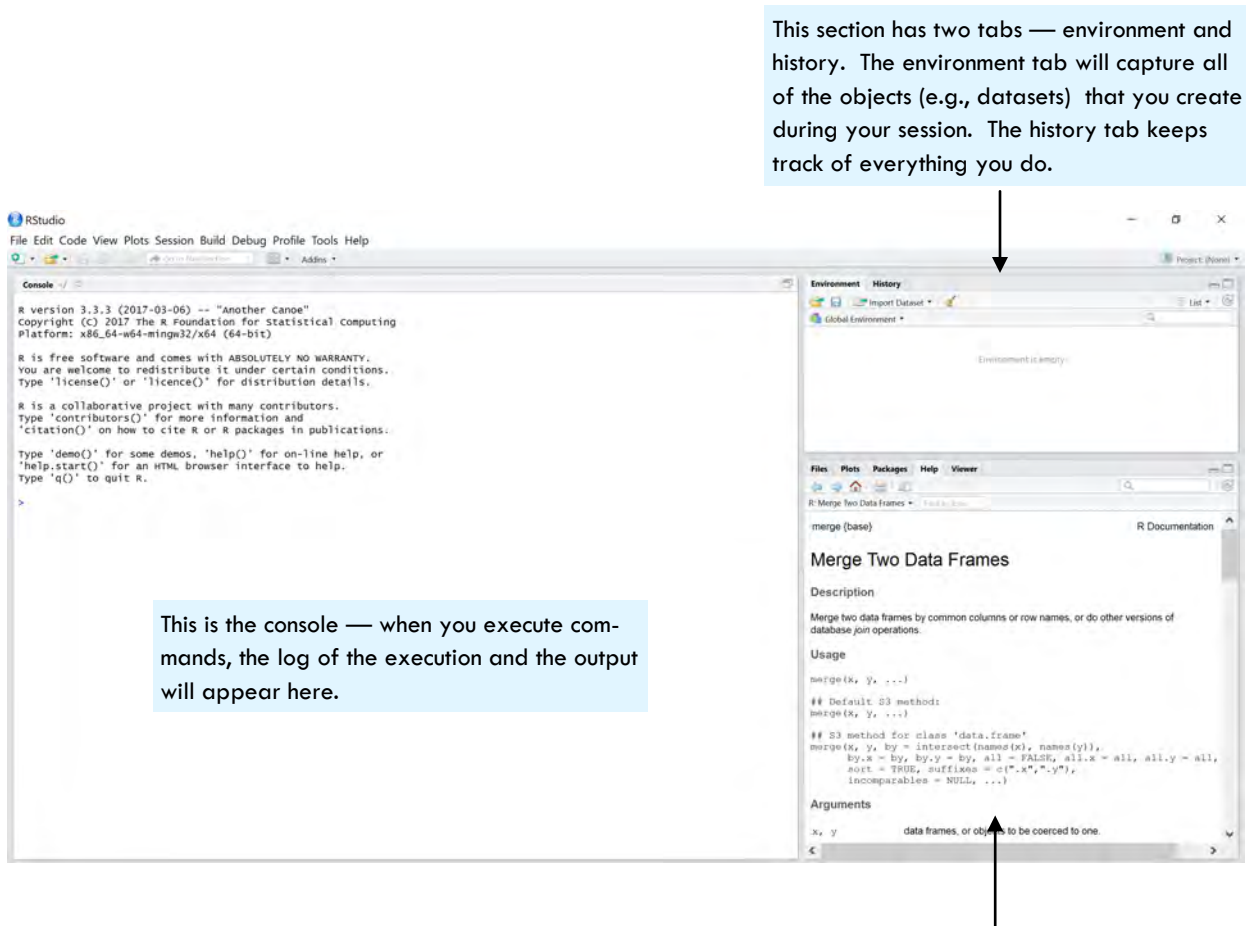
RStudio is an integrated development environment (IDE) for R. You can download it here: <http://www.rstudio.com/> (click on the download now button). R Studio is also updated regularly, but R Studio will prompt you to update and it's seamless to upgrade.

## Other Housekeeping Items

In order to manage and access shared files for PSY652 and PSY653, you need to create a folder to house all of these resources. First, on your hard drive (in a place where you can easily access) create a folder called RM. Prior to the start of each new unit, a Dropbox folder with the materials for the unit will be shared with you. When the folder appears in your Dropbox folder, copy the folder (e.g., Unit1), and paste it in your personal RM folder.

## Let's Open RStudio

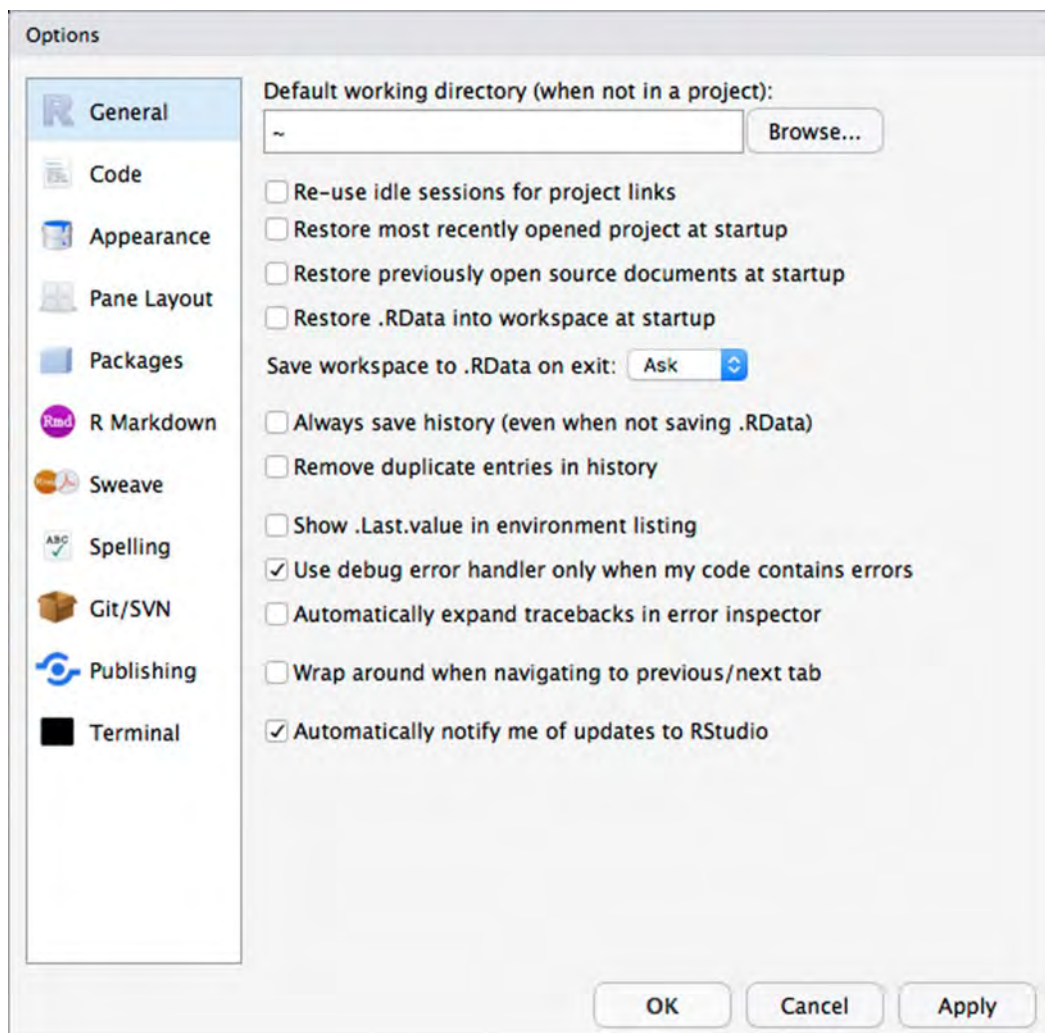
Double click on the R Studio shortcut on your desktop or choose it from the program's menu. RStudio uses R behind the scenes, so you do not need to open R.



## Configure R Studio

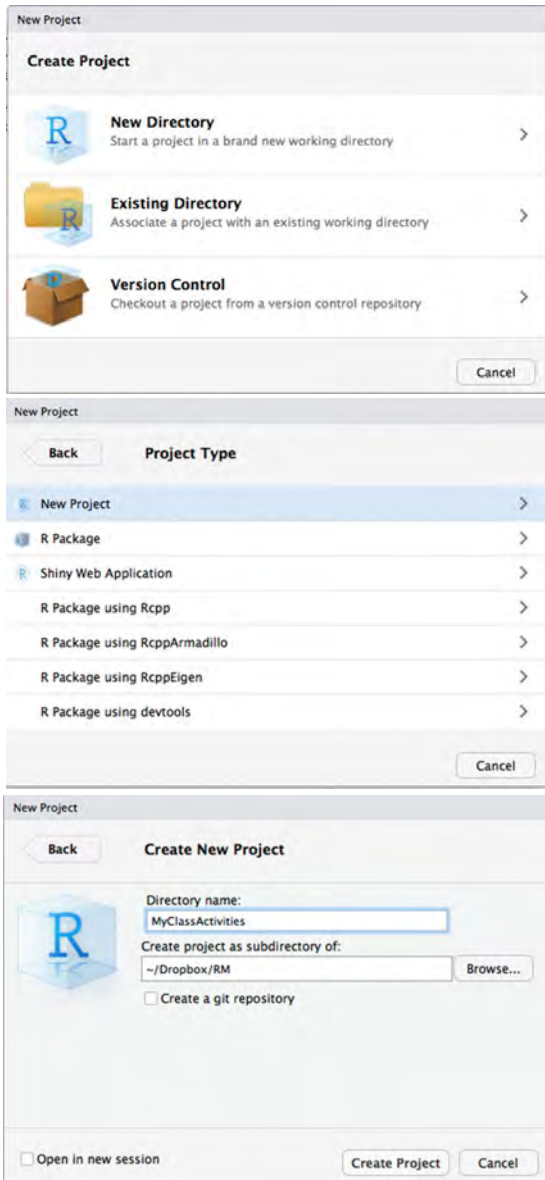
We need to configure RStudio to our liking. Here are my recommendations for configuring RStudio:

Click on **TOOLS > GLOBAL OPTIONS**. Then make the checked boxes and dropdown menus equivalent to those below.



## Starting a New Project

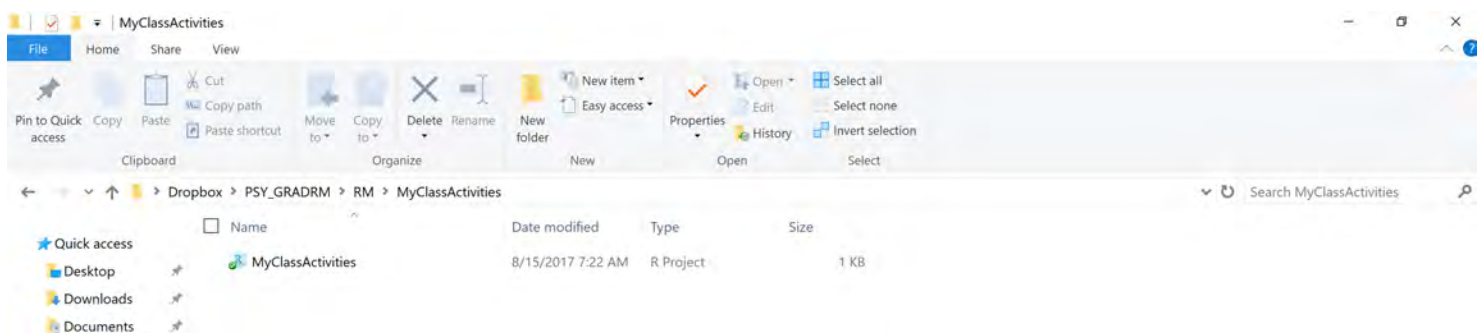
When you approach a new project, the first step is to create a project folder. To create a new project Click File > New Project, then:



Click New Directory

Click New Project

Name the directory MyClassActivities, and indicate that it will be stored in the dropbox directory you created on the previous page. Then click “Create Project” — this will create a folder called MyClassActivities, and a R Project file of the same name. All of the files associated with this new project will be stored in this folder. When you begin working on this project, you will either open the project (FILE > OPEN PROJECT) or double click on the project file (look for Type=R Project) from the file explorer (see below).

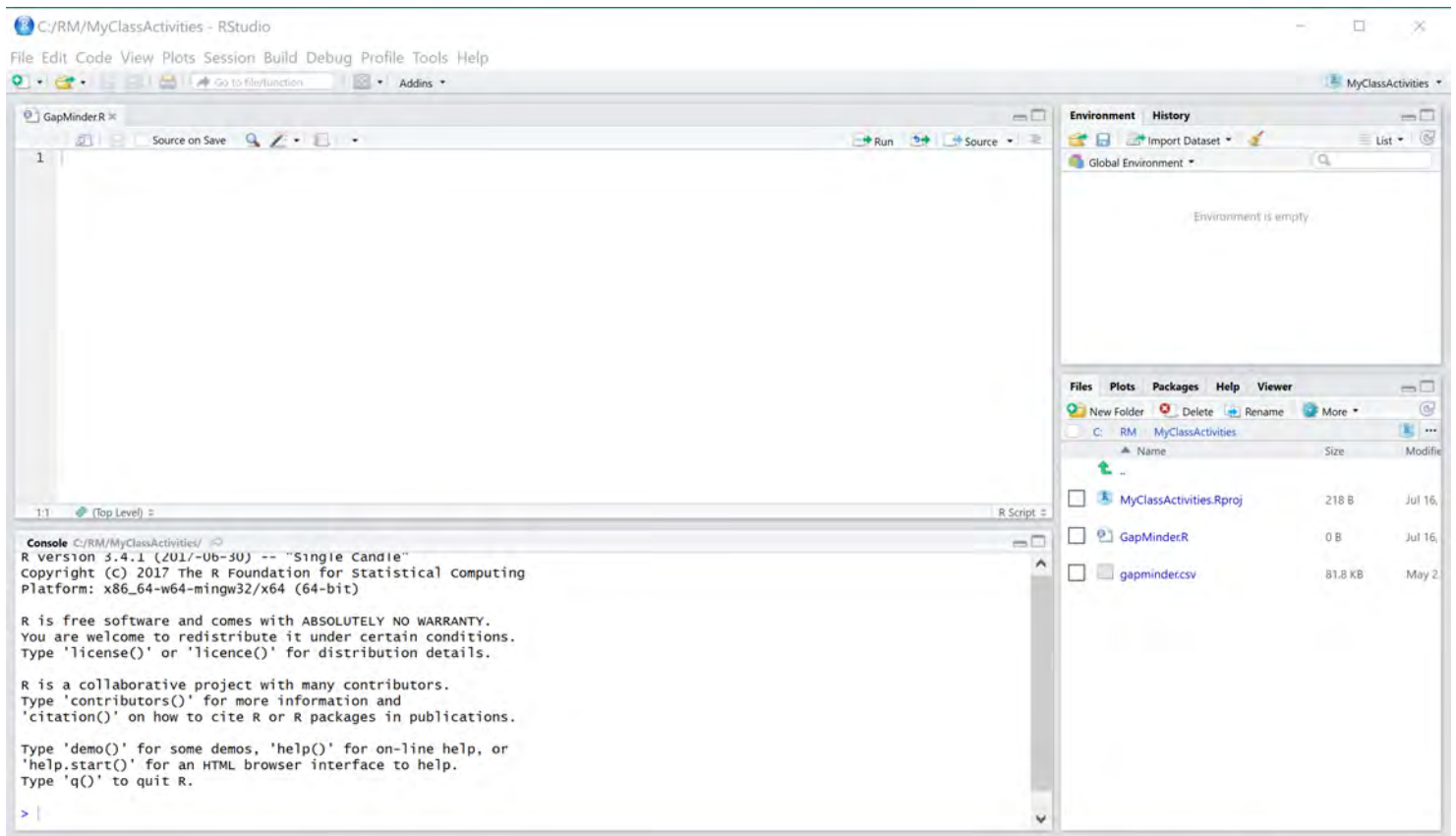


## Creating a R Script

Once a new project is created, you can begin creating files for the project. The first thing we need to do is put some data into the project folder. In the Unit1 folder that you copied over from Dropbox (and put in your RM folder) there is a csv file called gapminder.csv. Please copy that file into your MyClassActivities folder.

Now, go back to RStudio. To begin, we will use R through a R script. Click FILE > NEW FILE > R Script. This will open a new script. Let's save this script. Click on FILE > SAVE AS, and name the script GapMinder, since this will be a script that uses the gapminder dataset.

Notice that you now have an additional pane in RStudio, the R script. This is where we will type commands that will then be submitted to R for processing.



Notice that all of the files in our MyClassActivities project are listed in the files tab.



## Add a Package

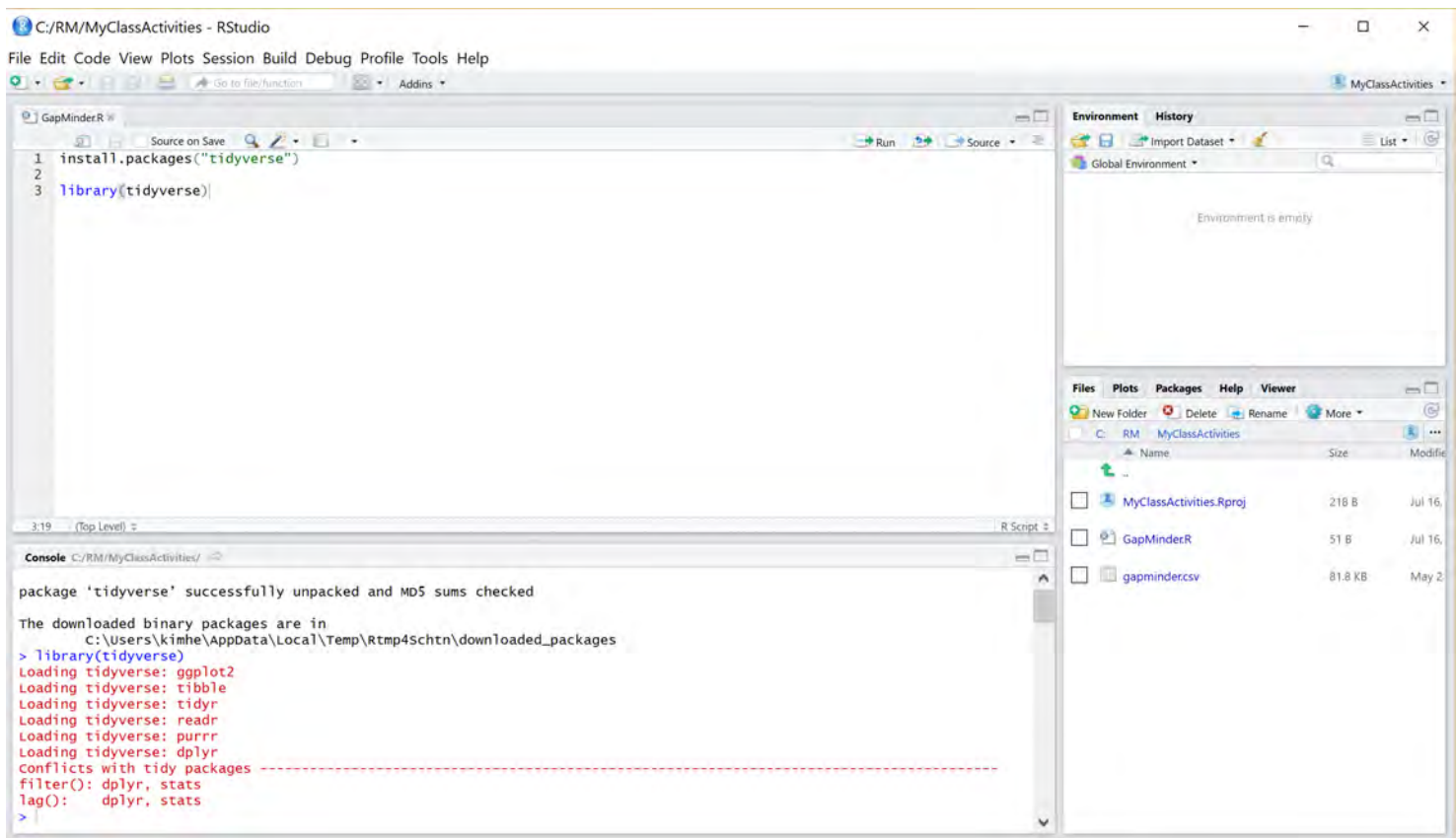
When we loaded R onto our machines we loaded base R. The true power of R is in the add on packages. These packages consist of a collection of functions that extend the capability of base R. Over the course of the semester, we will work with a variety of packages.

We will begin with a suite of packages called the tidyverse. The tidyverse is maintained by Hadley Wickham, the Chief Scientist at RStudio.

To install a package, you use the `install.packages` function, as follows:

`install.packages("package_name")` — where `package_name` is the package you desire to install. To execute code, highlight the code and then click the Run button. Once a package is installed, when you seek to use it in a R work session, you use the `library` function, as follows: `library(package_name)`.

```
install.packages("tidyverse")  
library(tidyverse)
```



If the package installed correctly, you will see the top message in the console — that is, that the package was successfully unpacked and checked. You only need to install a package once (although the package will need to be reinstalled when you download a new version of R and occasionally updated using `update.packages("package_name")`). You can either delete the `install.packages` line after you install this package, or you can put a `#` in front of it, which R will then ignore the line of code the next time you run this script. The `library` command, on the other hand, will need to be executed each session for which that package will be used. This calls the package into working memory for you to use. I typically put the `library` command line that calls in all of the packages that I will use in a particular session at the very top of my script. R is case-sensitive!!!

## Import a Dataset

To work with a dataset in R, we need to first import it. We will typically work with csv files in this course, so let's start by importing a dataset called `gapminder.csv`. It is the file you just copied into your new `MyClassActivities` folder.

The `gapminder` dataset is provided by an organization called `Gapminder.org`. It contains data on 142 countries. For each country, the dataset provides values for life expectancy, GDP per capita, and population — with these data available every five years from 1952 to 2007.

The `gapminder` dataset has 1704 rows and 6 variables:

- **country:** character variable with 142 levels
- **continent:** character variable with 5 levels
- **year:** numeric variable that ranges from 1952 to 2007 in increments of 5 years
- **lifeExp:** numeric variable that represents life expectancy at birth, in years
- **pop:** numeric variable that represents population
- **gdpPercap:** numeric variable that represents GDP per capita

```
gm <- read_csv(file="gapminder.csv")
```

```
names(gm)
str(gm)
head(gm)
```

In the code above, “names”, “str”, and “head” are functions. The function called `names` will list the names of all of the variables in the dataset. The function called `str` will give you information about the structure of the dataset. The function called `head` will print out the first few rows. In the first three functions, there is only one argument included (the dataset name). As we employ more complex functions, several arguments, separated by commas, will be used. For an example of this, enhance `head` by changing this to:

`head(gm, n=10)`. Notice that `n=10` is an additional argument which requests the top 10 rows, rather than the default 6.

```
> names(gm)
[1] "country" "continent" "year" "lifeExp" "pop" "gdpPercap"
> str(gm)
Classes 'tbl_df', 'tbl' and 'data.frame': 1704 obs. of 6 variables:
 $ country : chr "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
 $ continent: chr "Asia" "Asia" "Asia" "Asia" ...
 $ year : int 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
 $ lifeExp : num 28.8 30.3 32 34 36.1 ...
 $ pop : int 8425333 9240934 10267083 11537966 13079460 14880372 12881816
 $ gdpPercap: num 779 821 853 836 740 ...
> head(gm)
# A tibble: 6 x 6
  country continent year lifeExp pop gdpPercap
  <chr> <chr> <int> <dbl> <int> <dbl>
1 Afghanistan Asia 1952 28.801 8425333 779.4453
2 Afghanistan Asia 1957 30.332 9240934 820.8530
3 Afghanistan Asia 1962 31.997 10267083 853.1007
```

When you begin by creating or opening a R Project, R knows to look for the datafile in the project folder (e.g., `MyClassActivities` folder). Alternatively, you can specify the full path if you want to point to a different folder (e.g., `C:/other_data/datafile.csv`).

We use the `read_csv` function to read in the csv file. By listing `gm`, and then the assignment operator (`<-`), we are telling R to read in the `gapminder` csv file and name it `gm`. Notice that after importing this dataset, an object called `gm` now appears in the environment tab of RStudio, under Data. Double click on the `gm` dataframe in the environment tab and you can view it.

	country	continent	year	lifeExp	pop	gdpPercap
1	Afghanistan	Asia	1952	28.801	8425333	779.4453
2	Afghanistan	Asia	1957	30.332	9240934	820.8530
3	Afghanistan	Asia	1962	31.997	10267083	853.1007
4	Afghanistan	Asia	1967	34.020	11537966	836.1971
5	Afghanistan	Asia	1972	36.088	13079460	739.9811

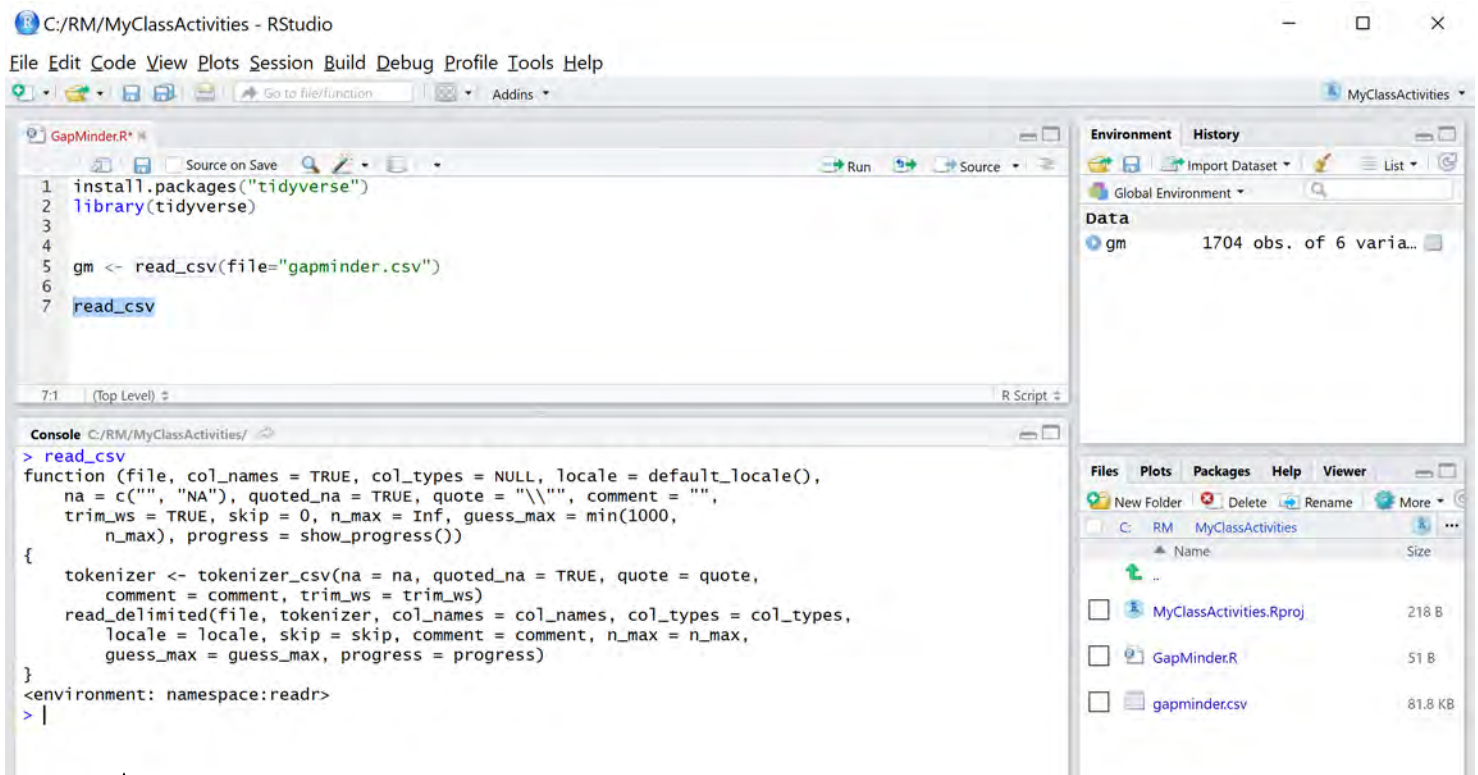
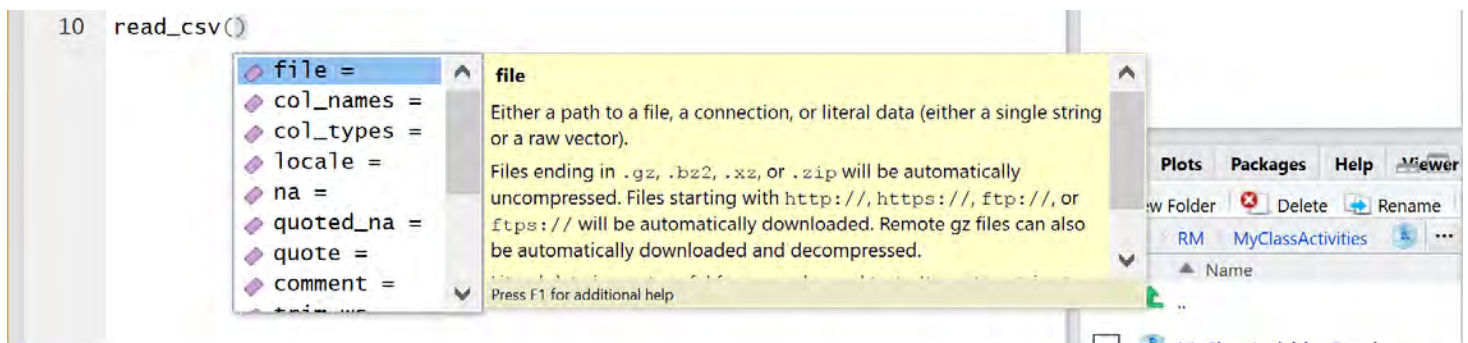
\*You will see R refer to real numbers as doubles (dbl). R will call text variables `chr` for character, and numerical variables without decimal places `int` for integer.



## A Bit on Functions

We just used a series of functions, for example, `read_csv`, `names`, `str`, etc. R is made up of millions of functions. A function is piece of code written to carry out a set of directions. A function (e.g., `read_csv`) will be followed by round brackets (i.e., parentheses) — inside the round brackets you can specify the arguments (i.e., options) of the function. In R studio, you can see the arguments of the function and get some help by typing the name of the function, the first round bracket and then hit tab (see first screen shot). You can also use the help tab in the lower right pane of RStudio to search for a function. Over the course of the semester, we will work with many functions and explore the most useful arguments that are possible for each.

You can see the code that makes up the function by typing the name of the function, highlight it, and click Run (see second screen shot). This will print out the code that makes up the function in the console.



Note that you will likely want to regularly clear the console. To do this, click EDIT > CLEAR CONSOLE

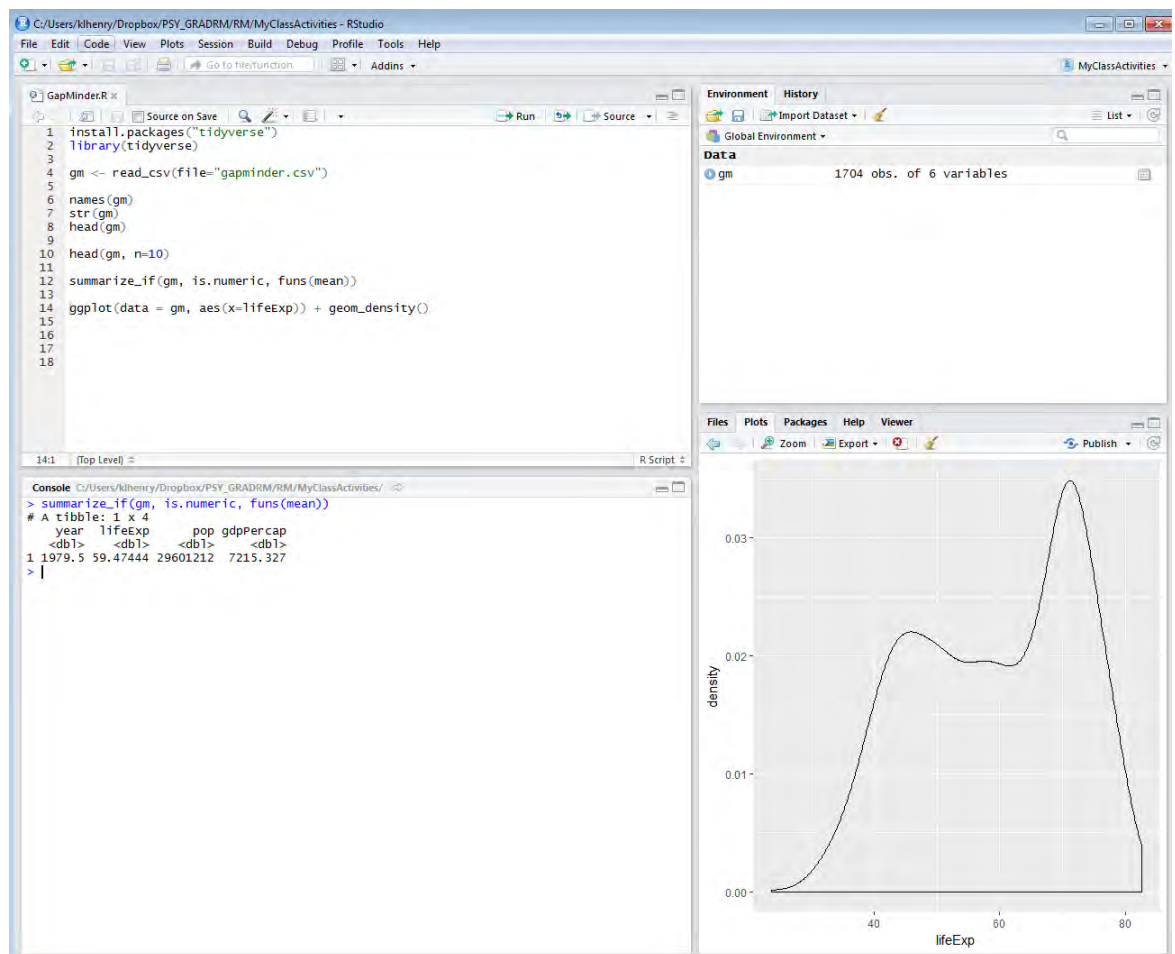
## Let's Submit Code to R

Over the coming weeks, we will work up to learning how to understand and write code, but for now, we'll just take a look at how code is submitted. First, let's get the mean of all of the numeric data in the gapminder dataframe.

```
summarize_if(gm, is.numeric, list(mean))
```

Second, let's create a simple plot to present the distribution of life expectancy.

```
ggplot(data = gm, aes(x=lifeExp)) + geom_density()
```



The results from the first line of code print out in the console, we see the mean of each of the numeric variables in the dataframe.

The results of the second line produce a plot, and the plot is displayed in the plot window of the lower right pane of RStudio.

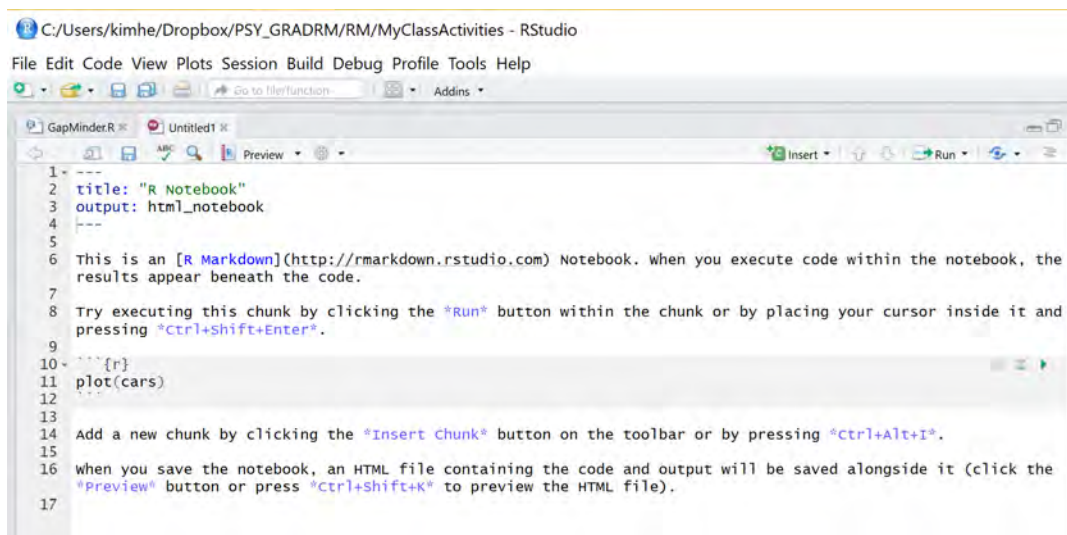
Now that we are done, let's save the R script. Notice that the name will change from RED to BLACK, indicating that all edits to the file have been saved. At this time, it is safe to exit. Please exit out of RStudio.

## R Notebooks

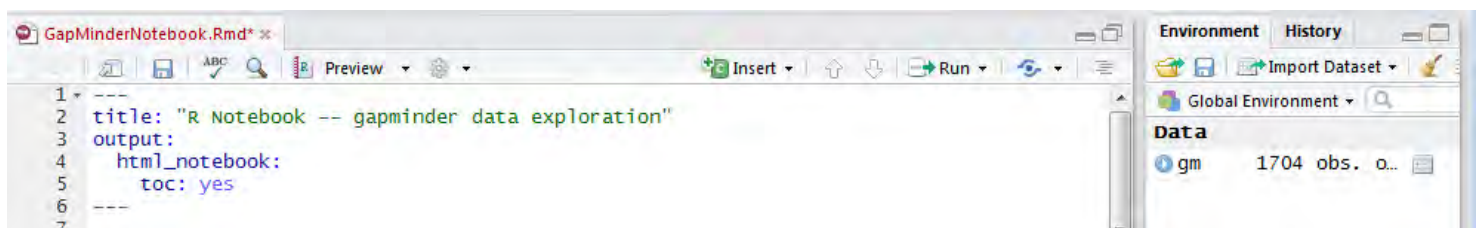
We just worked with R via a R script. In 2016, RStudio released a new way of managing analysis projects called R Notebooks. It's a great way of integrating R syntax and output, and creating slick and reproducible reports. This is the primary way in which we will interact with R in this course.

To begin, open your R project created earlier — FILE > OPEN PROJECT, and navigate to your MyClassActivities project. Now, start a R Notebook, click FILE > NEW FILE > R NOTEBOOK.

When you open up a new R NOTEBOOK, it contains some pre-created components to help get you started.



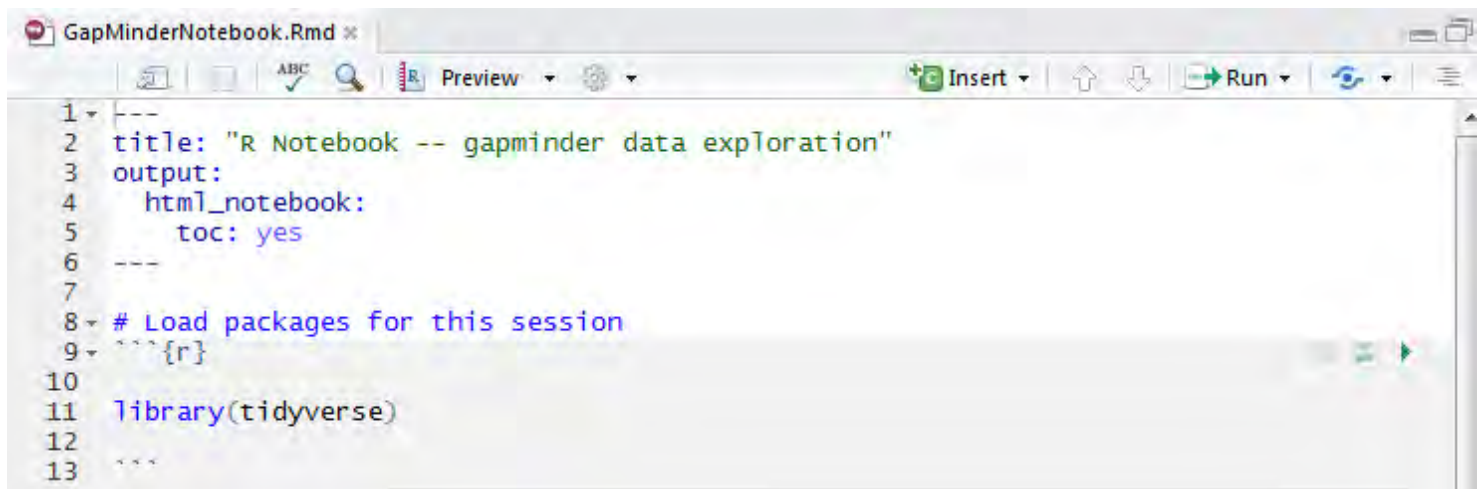
Let's begin to modify this. First, click on the gear icon, then click OUTPUT OPTIONS. On the first tab, you can request a table of contents (TOC). Next, add a more informative title (R Notebook -- gapminder data exploration). Save the notebook as GapminderNotebook in your current project folder. Note that it is best to not include spaces in the names of any files you use (e.g., script names, notebook names, datafile names).



## **R Notebooks — Load the tidyverse package**

The gray part just below this section label is called a R code chunk. Think of this as a mini R script. In this first code chunk, we will load the libraries needed. Let's call this code chunk "Load packages for this session." Note that the hashtag outside of a code chunk has a special role — it will allow us to build our TOC. One # is the first level header, two # (##) is a second level header, etc.

Now, let's load tidyverse using the library statement. Click the green arrow on the upper right side of the code chunk. This will run the contents of the code chunk.



The screenshot shows the RStudio interface for a notebook titled "GapMinderNotebook.Rmd". The code editor displays the following R code:

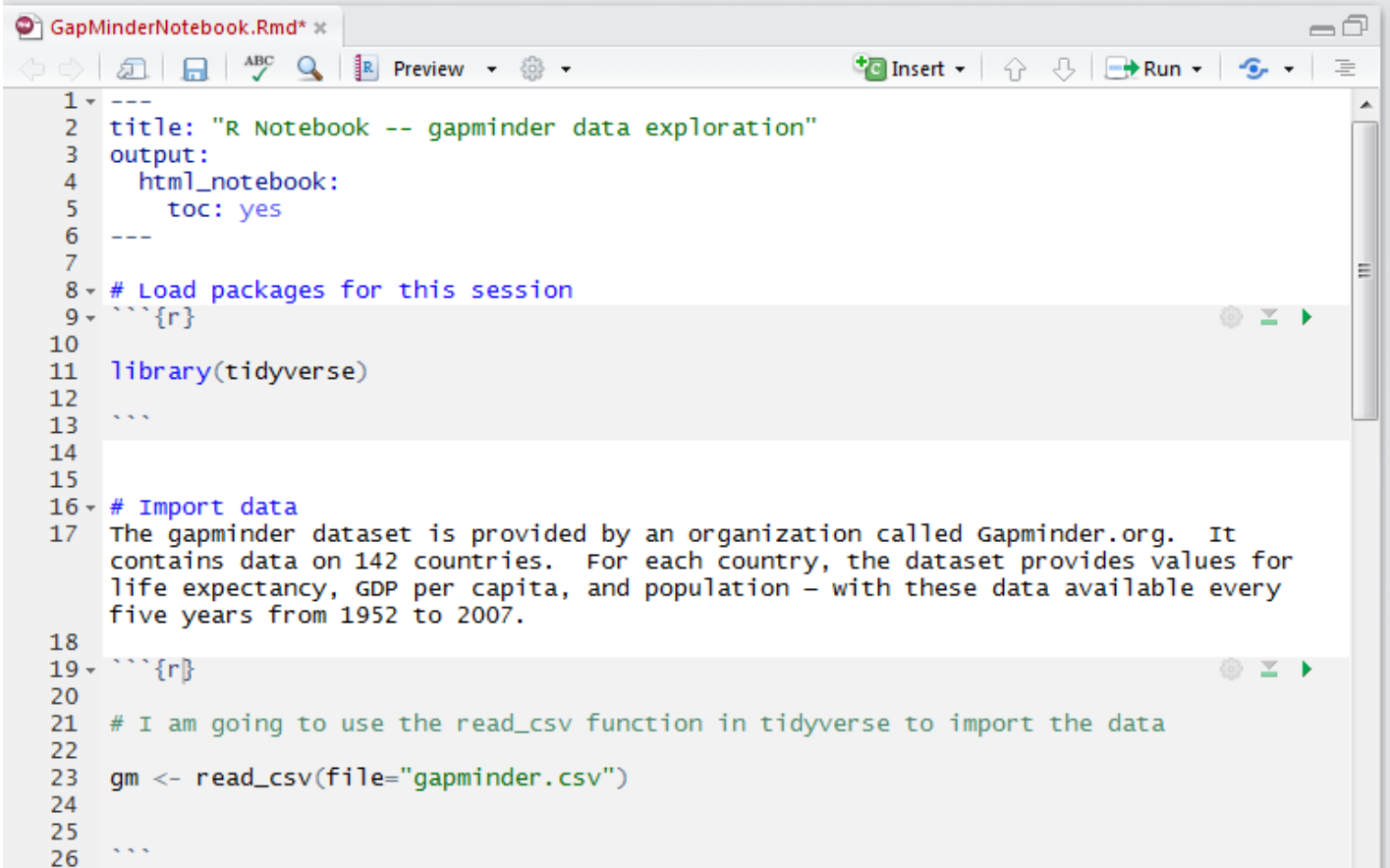
```
1 ---
2 title: "R Notebook -- gapminder data exploration"
3 output:
4   html_notebook:
5     toc: yes
6 ---
7
8 # Load packages for this session
9 ```{r}
10
11 library(tidyverse)
12
13 ```
```

The code is organized into a code chunk. The first part is a YAML header for the notebook, followed by a comment "# Load packages for this session" and an R code block enclosed in triple backticks. The R code block contains the `library(tidyverse)` statement. The interface includes a toolbar with icons for inserting code, running the chunk, and other notebook functions.

## R Notebooks — Import Data

Now, let's add another R code chunk to import the data. First, click INSERT (at top of pane) > R. This will make a new code chunk. Let's name this code chunk "Import data" - remember to put a # in front so R knows it's a first level header and will populate in our TOC.

Note that text not placed after a hashtag outside of a code chunk is a good place to describe what you're doing, provide more information to the reader, interpret results, etc. As was the case in the regular R script, if you want to make notes **inside** a code chunk, then always use the hashtag.



```
1 ---
2 title: "R Notebook -- gapminder data exploration"
3 output:
4   html_notebook:
5     toc: yes
6 ---
7
8 # Load packages for this session
9 ```{r}
10
11 library(tidyverse)
12
13 ```
14
15
16 # Import data
17 The gapminder dataset is provided by an organization called Gapminder.org. It
18 contains data on 142 countries. For each country, the dataset provides values for
19 life expectancy, GDP per capita, and population – with these data available every
20 five years from 1952 to 2007.
21
22 ```{r}
23
24 # I am going to use the read_csv function in tidyverse to import the data
25
26 gm <- read_csv(file="gapminder.csv")
27
28 ```
```



## R Notebooks — Create Output

Now, let's add another R code chunk to get some basic descriptive statistics. First, click INSERT (at top of pane) > R. Let's name this code chunk "Summarize the variables" - remember to put a # in front so R knows it's a first level header and will populate in our TOC. Inside the code chunk, let's ask for a summary of the variables in the dataframe. We do this with the summary function. Once complete, click the green arrow to execute the code chunk.

```
# Summarize the variables
---{r}

summary(gm)
```

country	continent	year	lifeExp	pop
Length:1704	Length:1704	Min. :1952	Min. :23.60	Min. :6.001e+04
Class :character	Class :character	1st Qu.:1966	1st Qu.:48.20	1st Qu.:2.794e+06
Mode :character	Mode :character	Median :1980	Median :60.71	Median :7.024e+06
		Mean :1980	Mean :59.47	Mean :2.960e+07
		3rd Qu.:1993	3rd Qu.:70.85	3rd Qu.:1.959e+07
		Max. :2007	Max. :82.60	Max. :1.319e+09

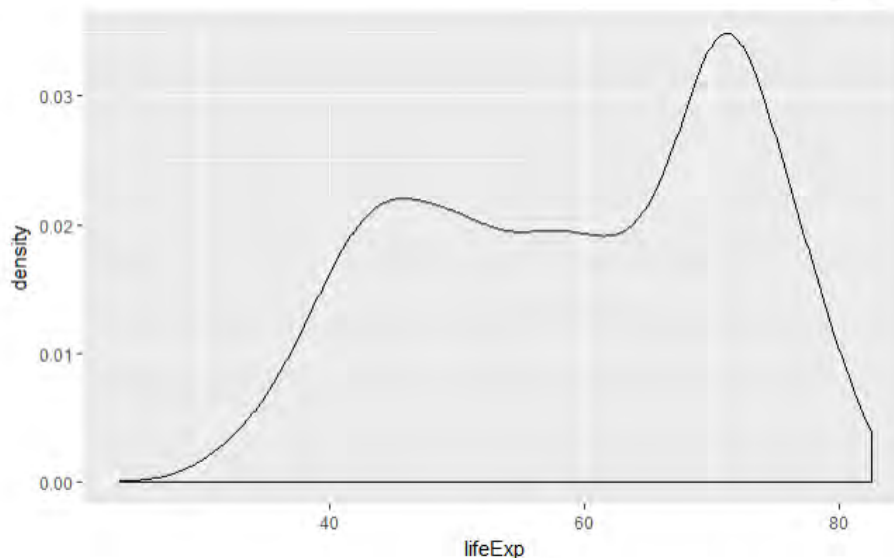
  

```
gdpPercap
Min. : 241.2
1st Qu.: 1202.1
Median : 3531.8
Mean : 7215.3
3rd Qu.: 9325.5
Max. :113523.1
```

Let's add one more code chunk to create a simple plot. Again, click INSERT > R. Label the code chunk as below and type the commands to create a density plot of one of the variables in the dataframe.

```
# Create a plot
---{r}

ggplot(data = gm, aes(x=lifeExp)) + geom_density()
```





## Create the Full Notebook

Now that we have all of the analyses complete, let's restart R and run the entire file. Do this by clicking RUN > RE-START R AND RUN ALL CHUNKS. This is going to restart R and run your entire sequence from top to bottom. This is an important step because we want to ensure all code is in the correct sequence and executes without error. This is what helps to ensure your code is REPRODUCIBLE. Once you verify that everything is in order, click PREVIEW > PREVIEW NOTEBOOK. This will cause a pop up file of your notebook to appear. The notebook output is also automatically saved in your project folder. It will have the same name as your notebook input file, but with a .nb extension.

**R Notebook – gapminder data exploration**

- Load packages for this session
- Import data
- Summarize the variables

### Load packages for this session

```
library(tidyverse)
```

Loading tidyverse: ggplot2  
Loading tidyverse: tibble  
Loading tidyverse: tidyr  
Loading tidyverse: readr  
Loading tidyverse: purrr  
Loading tidyverse: dplyr  
Conflicts with tidy packages -----  
filter(): dplyr, stats  
lag(): dplyr, stats

### Import data

The gapminder dataset is provided by an organization called Gapminder.org. It contains data on 142 countries. For each country, the dataset provides values for life expectancy, GDP per capita, and population — with these data available every five years from 1952 to 2007.

```
gm <- read_csv(file="gapminder.csv")
```

Parsed with column specification:

```
cols(
  country = col_character(),
  continent = col_character(),
  year = col_integer(),
  lifeExp = col_double(),
  pop = col_integer(),
  gdpPercap = col_double()
)
```

### Summarize the variables

```
summary(gm)
```

country	continent	year	lifeExp
Length:1704	Length:1704	Min. :1952	Min. :23.60
Class :character	Class :character	1st Qu.:1966	1st Qu.:48.20
Mode :character	Mode :character	Median :1980	Median :60.71
		Mean :1980	Mean :59.47
		3rd Qu.:1993	3rd Qu.:70.85
		Max. :2007	Max. :82.60
pop	gdpPercap		
Min. :6.001e+04	Min. : 241.2		
1st Qu.:2.794e+06	1st Qu.: 1202.1		
Median :7.024e+06	Median : 3531.8		
Mean :2.960e+07	Mean : 7215.3		
3rd Qu.:1.959e+07	3rd Qu.: 9325.5		
Max. :1.319e+09	Max. :113523.1		

Let's end the session by saving the Notebook file, and then close RStudio.

## **Tips for R Notebooks**

1. Think carefully about headings and subheadings and use the hashtags to populate the table of contents. As your notebook grows, the table of contents will become very important in helping you to find sections of your report.
2. Underneath headings (but not in the code chunk), you can have text to describe why you did something or interpret results.
3. Try to make each code chunk create only one piece of output, this makes reading the outputted notebook easier.
4. Before finalizing the report, always “Restart R and Run All Code Chunks”